BAB II
TINJAUAN PUSTAKA

2.1 Penelitian Terkait

Adapun jurnal atau penelitian yang berhubungan dengan laporan skripsi ini antara lain :

Eko Prasetyo Adi Sutrisno, Program Aplikasi GPS Untuk Mencari Lokasi Dan Jarak SPBU Di Tangerang Selatan Dengan Peta Dan Augmented Reality Camera-View Pada Perangkat Bergerak Berbasis Android. Aplikasi ini menampilkan lokasi stasiun pengisian bahan bakar umum menggunakan peta digital dengan memanfaatkan layanan *GPS* yang dimiliki oleh setiap perangkat android yang beredar dan juga ditambahkan dengan teknologi *Augmented Reality* yang membuat aplikasi ini semakin menarik[1].

Artha Eka Darmawan, Nerfita Nikentari, ST, M.Cs dan Martaleli Bettiza S.Si, M.Sc, Sistem Informasi Geografis Stasiun Dan Pengisian Bahan Bakar Umum Di Kota Batam. Sistem ini membahas tentang bagaiamana membuat sebuah sistem informasi geografis berbasis web yang dipadukan dengan *Quantum GIS* dengan menggunakan metode *waterfall*, sistem informasi ini menampilkan peda pada halaman web dengan memanfaatkan *alov map* untuk mempermudah user dalam membaca peta yang ditampilkan[2].

Fajri Mustaqim, Sistem Informasi Geografis Jalur Trayek Bus Di Kota Semarang menggunakan Arcview GIS. Di sistem informasi ini peneliti masih menggunakan teknologi Arcview GIS dimana dalam teknologi tersebut peneliti harus menyediakan peta sendiri dan tentu saja dengan begitu peta yang dihasilkan kurang mendetail dan kelihatan kurang menarik dibandingkan peta yang sudah disediakan oleh google[3].

Berdasarkan dari beberapa referensi yang telah diuraikan sebelumnya maka penuilis berusaha untuk membuat sebuah sistem informasi tentang lokasi Stasiun Pengisian Bahan Bakar Umum yang terdapat di Kota Jepara dan Kudus memanfaatkan Google Maps API yang di dukung dengan teknologi Node-JS dan MongoDB. Para pengguna sistem ini diharapkan dapat dengan mudah mendapatkan akses informasi mengenai lokasi terdekat Stasiun Pengisian Bahan Bakar Umum

tanpa harus menggunakan salah satu platform sistem operasi tertentu, dan dapat diakses melalui gadget mereka yang dapat terhubung dengan koneksi internet tentunya.

2.1 Landasan Teori

2.2.1 Pengertian Sistem Informasi Geografis

2.1.1.1 Sistem

Menurut Gerald. J. Dkk, di dalam Al-Bahra Bin Ladjamudin(2005), sistem adalah suatu jaringan kerja dari sebuah prosedur-prosedur yang saling berhubungan, berkeumpul bersama-sama untuk melakukan suatu kegaitan atau untuk menyelesaikan suatu sasaran tertentu [4].

2.1.1.2 Informasi

Informasi menurut Al-Bahra Bin Ladajamudin (2005), adalah data yang diolah menjadi bentuk yang berguna dan mempuyai suatu arti bagi penerima informasi. Kegunaan informasi adalah unutk mengurangi ketidakpastian di dalam proses pengambilan keputusan tentang suatu keadaan [4].

2.1.1.3 Sistem Informasi

Menurut Al-Bahra Bin Ladjamudin (2005), Sistem Informasi dapat di definisikan sebagai berikut:

1. Suatu sistem yang dibuat oleh manusia yang terdiri dari komponen-komponen dalam organisasi untuk mencapai suatu tujuan, yaitu menyajikan informasi.
2. Sekumpulan prosedur organisasi yang ada pada saat dilaksanakan akan memberikan informasi bagi penngambil keputusan dan atau untuk mengendalikan organisaisi.
3. Suatu sistem dalam suatu organisasi yang mempertemukan kebutuhan pengoahan trasaksi, mendukung operasi, bersifat managerial, dan kegiatan strategi dari suatu

organisasi dan menyediakan pihak luar tertentu dengan laporan-laporan yang diperlukan [4].

2.2.1.4 Geografis

Pengertian Geografi pada umumnya adalah ilmu pengetahuan yang mempelajari tentang lokasi serta persamaan dan perbedaan variasi keruangan atas fenomena fisik dan manusia diatas permukaan bumi. Geografi lebih dari sekedar Kartografi (studi tentang peta). Geografi tidak hanya menjawab apa dan dimana yang ada diatas muka bumi tetapi juga diartikan pada lokasi pada ruang. Dengan kata lain, Geografi adalah ilmu pengetahuan yang menggambarkan segala sesuatu yang ada di permukaan bumi.[1]

2.2.1.5 Sistem Informasi Geografis

Sistem Informasi Geografis atau dalam Bahasa Inggris Geographic Information System (GIS) merupakan sistem informasi khusus yang mengelola data yang memiliki informasi spasial maupun non-spasial, untuk mendukung pengambilan keputusan dalam perencanaan dan pengelolaan suatu wilayah [5].

2.2.1.6 Subsistem Sistem Informasi Geografis

Sistem Informasi Geografi dapat diuraikan menjadi beberapa subsistem, yaitu:

1. Data *Input*

   Subsistem ini bertugas untuk mengumpulkan dan mempersiapkan data spasial dan atribut dari berbagai sumber. Subsistem ini pula yang bertanggung jawab dalam mengkonversi atau mentranformasikan format-format yang dapat digunakan oleh sistem informasi geografi.

[1] http://dedigeografi.blogspot.com/2012/03/istilah-geografi-berasal-dari-bahasa.html (diakses pada tanggal 9 Maret 2015 pada jam 19.00 WIB)

2. Data *Output*

   Subsistem ini menampilkan atau menghasilkan keluaran seluruh atau sebagian basis data baik dalam bentuk *softcopy* maupun dalam bentuk *hardcopy* seperti tabel, grafik, peta, dan lain-lain.

3. Data Manajemen

   Subsistem ini mengorganisasikan baik data spasial maupun atribut ke dalam sebuah basis data sedemikian rupa sehingga mudah dipanggil, diperbaharui, dan diperbaiki.

4. *Data Manipulation and Analysis*

   Subsistem ini menentukan informasi-informasi yang dapat dihasilkan oleh sistem informasi geografi. Selain itu, subsistem ini juga melakukan manipulasi dan pemodelan data untuk menghasilkan informasi yang diharapkan.[2]

2.2.1.7 Komponen Sistem Informasi Geografis

Komponen-komponen pendukung SIG terdiri dari lima komponen yang bekerja secara terintegrasi yaitu perangkat keras (hardware), perangkat lunak (software), data, manusia, dan metode yang dapat diuraikan sebagai berikut:

1. Perangkat Keras (Hardware).

   Perangkat keras SIG adalah perangkat-perangkat fisik yang merupakan bagian dari sistem komputer yang mendukung analisis goegrafi dan pemetaan. Perangkat keras SIG mempunyai kemampuan untuk menyajikan citra dengan resolusi dan kecepatan yang tinggi serta mendukung operasioperasi basis data dengan volume data yang besar secara cepat. Perangkat keras SIG terdiri dari beberapa bagian untuk menginput data, mengolah data, dan mencetak hasil proses. Berikut ini pembagian berdasarkan proses:

   a. Input Data: mouse, digitizier, scanner.
   b. Olah Data: harddisk, processor, RAM, VGA card.
   c. Output Data: plotter, printer, screening [5].

[2] http://coretan-kuliah-ku.blogspot.com/2012/10/sub-sistem-gissig.html (diakses pada tanggal 7 Maret 2015 pada jam 11.00 WIB)

2. Perangkat Lunak (Software).

   Perangkat lunak digunakan untuk melakukan proses menyimpan, menganalisa, memvisualkan data-data baik data spasial maupun non-spasial. Perangkat lunak yang harus terdapat dalam komponen software SIG adalah:

   a. Alat untuk memasukkan dan memanipulasi data SIG.
   b. Data Base Management System (DBMS).
   c. Alat untuk menganalisa data-data.
   d. Alat untuk menampilkan data dan hasil analisa [5].

3. Data.

Pada prinsipnya terdapat dua jenis data untuk mendukung SIG yaitu:

a. Data Spasial

   Data spasial adalah gambaran nyata suatu wilayah yang terdapat di permukaan bumi. Umumnya direpresentasikan berupa grafik, peta, gambar dengan format digital dan disimpan dalam bentuk koordinat x,y (vektor) atau dalam bentuk image (raster) yang memiliki nilai tertentu [5].

b. Data Non Spasial (Atribut)

   Data non spasial adalah data berbentuk tabel dimana tabel tersebut berisi informasi- informasi yang dimiliki oleh obyek dalam data spasial. Data tersebut berbentuk data tabular yang saling terintegrasi dengan data spasial yang ada [5].

4. Manusia.

Manusia merupakan inti elemen dari SIG karena manusia adalah perencana dan pengguna dari SIG. Pengguna SIG mempunyai tingkatan seperti pada sistem informasi lainnya, dari tingkat spesialis teknis yang mendesain dan mengelola sistem sampai pada pengguna yang menggunakan SIG untuk membantu pekerjaannya sehari-hari [5].

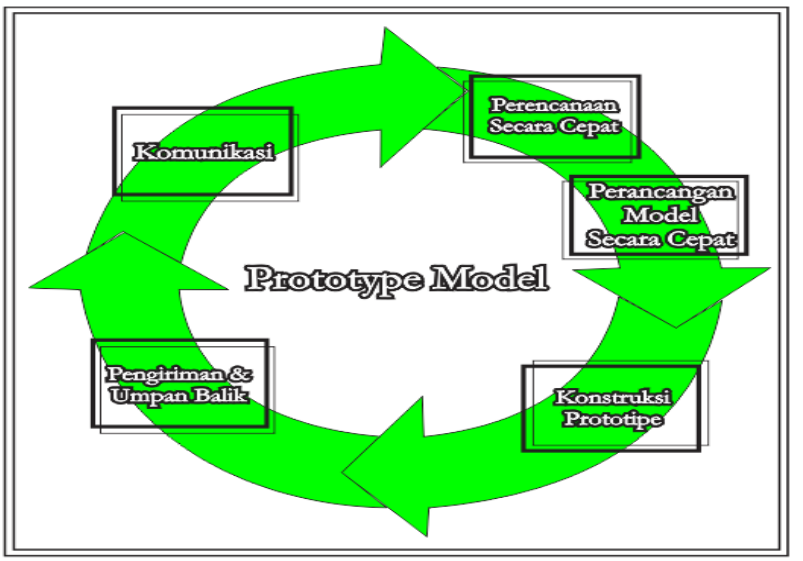
5. Metode.

Metode yang digunakan dalam SIG akan berbeda untuk setiap permasalahan. SIG yang baik tergantung pada aspek desain dan aspek realnya [5].

2.2.2 Metode Pengembangan Perangkat Lunak

Metode *Prototype* merupakan metode pengembangan perangkat lunak yang memodelkan dari sistem kerja suatu perangkat lunak yang belum lengkap dari pihak *user*. Para pengembang perangkat lunak melakukan koordinasi dan pertemuan-pertemuan yang secara intensif dengan *user* guna menampung informasi yang akan dijadikan dasar dalam perancangan perangkat lunak. Perubahan-perubahan dalam metode prototype ini dapat dilakukan berkali-kali sampai dicapai kesepakatan bentuk perangkat lunak yang akan dipakai.

Sering kali seorang pelanggan mendefinisikan serangkaian sasaran umum bagi perangkat lunak, tetapi tidak mengidentifikasi kebutuhan yang rinci untuk fungsi-fungsi dan fitur-fitur yang nantinya akan dimiliki perangkat lunak yang akan dikembangkan. Pengembang perangkat lunak mungkin merasa tidak pasti tentang efisiensi suatu algoritma yag akan digunakan dalam pengembangan perangkat lunak, atau merasa tidak pasti dengan kemampuan perangakat lunak yang dapat beradaptasi dengan sistem operasi yang akan digunakan, atau mungkin juga akan merasa tidak pasti dengan apa interaksi manusia dengan komputer atau perangkat yang akan digunanak. Dalam situasi seperti ini paradigma pembuatan *prototype* mungkin bisa menawarkan sebuah pendekatan yang baik [6]. Pemodelan *prototype* dapat dilihat pada gambar 2.1.



Gambar 2.1 : *Protoype Paradigm*

2.3 Perancangan Sistem

2.3.1 *Flowchart*

Flowchart adalah bagan-bagan yang mempunyai arus yang menggambarkan langkah-lanhkah penyelesaian suatu masalah. *Flowchart* merupakan cara penyajian dari suatu algoritma [4]. Ada dua macam *flowchart* yang menggambarkan proses dengan komputer, yaitu:

1. Sistem *Flowchart*

   Bagan yang memperlihatkan urutan proses dalam sistem dengan menunjukan alat media *input*, *output* serta jenis media penyimpanan dalam proses pengolahan data.
2. Program *Flowchart*

   Bagan yang memperlihatkan urutan instruksi yang digambarkan dengan simbol tertentu untuk memecahan masalah dalam suatu program.

Flowchart disusun dengan simbol. Simbol ini dipakai sebagai alat bantu menggambarkan proses di dalam program. Simbol-simbol yag digunakan dapat dibagi menjadi tiga kelompok, yaitu :

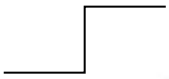
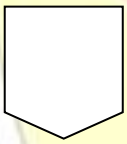
1. *Flow Directions Symbols*

   Flow Directions Symbols merupakan simbol penghubung atau alur antara poses satu ke proses yang lain. Macam-macam simbol penghubung akan dijelaskan pada tabel 2.1

Tabel 2.1 : *Flow Directions Symbols* [sumber: 4]

Simbol	Nama	Keterangan
	Arus atau Flow	Untuk menyatakan jalannya arus suatu proses

Tabel 2.1 (Lanjutan)

	*Communicatioin Link*	Untuk menyatakan bahwa adanya suatu transisi suatu data atu nformasi dari satu lokasi ke lokasi lainnya.
	*Connector*	Untuk menyatakan sambungan dari satu proses ke proses lainnya dalam halaman atau lembar yang sama.
	*Offline Connector*	Untuk menyatakan sambungan dari suatu proses ke proses lainnya dari halaman atau lembar yang berbeda.

2. *Processing Symbols*

   Processing Symbols merupakan simbol proses yang terdapat pada *flowchart*. Macam-macam simbol proses akan dijelaskan pada tabel 2.2.

Tabel 2.2 : *Processing Symbols* [sumber: 4]

Simbol	Nama	Keterangan
	*Offline Connector*	Untuk menyatakan sambungan dari suatu proses ke proses lainnya dari halaman atau lembar yang berbeda.
	*Manual*	Untuk menyatakan suatu tindakan (proses) yang tidak dilakukan oleh komputer (manual).
	*Decision atau Logika*	Untuk menunjukan suatu kondoso tertentu yang kana menghasilkan dua kemungkinan jawaban, ya atau tidak.
	*Predefined Procces*	Untuk menyatakan penyediaan tempat penyimpanan suatu pengolahan untuk memberi harga awal.
	*Terminal*	Untuk menyatakan permulaan atau akhir suatu program.

Tabel 2.2 (Lanjutan)

	*Keying Operation*	Untuk mrnyatakan segala jenis operasi yang diproses dengan menggunakan suatu mesin yang mempunyai keyboard
	*Off-line Storage*	Untuk menunjukan bahwa data dalam simbol ini akan disimpan ke suatu media tertentu.
	*Manual Input*	Untuk memasukan data secara manual dengan menggunakan online keyboard.

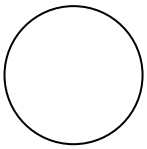
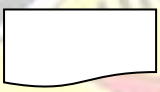
3. *Input-Output Symbols*

   Input-Output Symbols merupakan simbol input-output yang terdapat pada *flowchart*. Macam-macam simbol input-output akan dijelaskan pada tabel 2.3.

Tabel 2.3 : *Input-Output Symbols* [sumber: 4]

Simbol	Nama	Keterangan
	*Input - Output*	Untuk menyatakan proses input dan output tanpa tergantung dengan jenis peralatannya.

Tabel 2.3 (Lanjutan)

	*Punched Chart*	Untuk menyatakan input berasal dari kartu atau output ditulis ke kartu
	*Magnetic – tape Unit*	Untuk menyatakan input berasal dari pita magnetic atau output disimpan ke pita magnetic
	*Disk Storage*	Untuk menyatakan input berasal dari disk atau output disimpan ke disk.
	*Document*	Untuk mencetak laporan ke printer
	*Display*	Untuk menyatakan peralatan *output* yang digunakan berupa layar (video, komputer).

2.3.2 *Unified Modeling Language* (UML)

Unified Modeling Language (UML) adalah sebuah bahasa yang telah menjadi standar dalam industri untuk visualisasi, merancang dan mendokumentasikan sistem piranti lunak. UML menawarkan sebuah standar untuk merancang model sebuah sistem.

Dengan menggunakan UML kita dapat membuat model untuk semua jenis aplikasi piranti lunak, dimana aplikasi tersebut dapat berjalan pada piranti keras, sistem operasi dan jaringan apapun, serta ditulis dalam bahasa pemrograman apapun. Tetapi karena UML juga menggunakan *class* dan operation dalam konsep dasarnya, maka lebih cocok untuk penulisan piranti lunak dalam bahasa-bahasa berorientasi

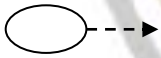
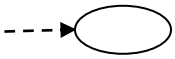
objek seperti C++, Java, C# atau VB.NET. Meskipun demikian, UML tetap dapat digunakan untuk modeling aplikasi prosedural dalam VB atau C.

Seperti bahsa-bahasa lainnya, UML mendefinisikan notasi dan *syntax* atau semantik. Notasi UML merupakan sekumpulan bentuk khusus untuk menggambarkan berbagai diagram piranti lunak. Setiap bentuk memiliki makna tertentu, dan UML *syntax* mendefinisikan bagaimana bentuk-bentuk tersebut dapat dikombinasikan. Notasi UML terutama diturunkan dari tiga notasi yang telah ada sebelumnya : *Grady Booch OOD (Object Oriented Design), Jim Rumbaugh OMT (Object Modeling Technique),* dan *Ivar Jacobson OOSE (Object-Oriented Software Engineering)*[7].

2.3.2.1 Use Case

Use case diagram atau *diagram use case* merupakan pemodelan untuk menggambarkan kelakuan (*behavior*) sistem yang akan dibuat. Dalam *use case* mendeskripsikan sebuah interaksi antara satau atau lebih aktor dengan sistem yang akan dibuat. Dengan pengertian yang cepat, *diagram use case* digunakan untuk mengetahui fungsi apa saja yang ada di dalam sebuah sistem dan siapa saja yang berhak menggunakan fungsi-fungsi tersebut. Terdapat beberapa simbol dalam menggambarkan *diagram use case*, yaitu *use case*, aktor dan relasi. Hal yang perlu diingat mengenai *diagram use case* adalah *diagram use case* bukan menggambarkan tampilan antar muka (*user interface*), arsitektur dari sistem, kebutuhan non-fungsional, dan tujuan performansi. Sedangkan untuk penamaan *use case* adalah nama didefinisikan sesederhana mungkin, dapat dipahami dan menggunakan kata kerja. Berikut adalah Simbol-simbol pada *use case* seperti pada tabel 2.4 [7].

Tabel 2.4 : Simbol *Use Case* [Sumber**: 7**]

Simbol	Nama	Deskripsi
	*Use Case*	Fungsionalitas yang disediakan sistem sebagai unit-unit yang saing bertkar pesan antar unit atau aktor ; biasanya dinyatakan dengan menggunakan kata kerja diawal frase nama *use case*
	Aktor	Orang, proses, atau sistem lain yang berinteraksi dengan sistem informasi yang akan dibuat diluar sistem informasi yang akan dibuat itu sendiri, jadi walaupun simbol dari aktor adalah gambar orang tapi aktor belum tentu merupakan orang ; biasanya dinyatakan menggunakan kata benda diawal frase nama aktor
	Asosiasi atau *Association*	Komunikasi antar aktor dan *use case* yang berpartisipasi pada *use case* atau *use case* memiliki interaksi dengan aktor.
	*Extend*	Relasi *use case* tambahan ke sebuah *use case* dimana *use case* yang ditambahkan dapat berdiri sendiri walau tanpa *use case* tambahan itu ; mirip dengan prisnsip *inheretannnce* pada pemrograman berorientasi objek ; biasanya *use case* tambahan memiliki nama depan yang sama dengan *use case* yang ditambahkan, arah panah menunjuk pada *use case* yang dituju.
	*Include*	Relasi *use case* tambahan ke sebuah *use case* dimana *use case* yang ditambahakan memerlukan *use case* ini yang menjalankan fungsinya atau sebagai syarat dijalankan *use case* ini. Ada dua sudut pandang yang cukup besar mengenai include di *use case*, *include* berarti *use case* yang ditambahakan akan selalu dipanggil saat *use case* tambahan dijalankan.

2.3.2.2 Class Diagram

Diagram kelas atau *class diagram* mengambarkan struktur sistem dari segi pendefinisian kelas-kelas yang akan dibuat untuk membangun sietem. Kelas memiliki apa yang disebut atribut dan metode atau operasi.

a. Atribut merupakan variabel-variabel yang dimiliki oleh suatu kelas
b. Atribut mendeskripsikan properti dengan sebaris teks didalam kotak kelas tersebut
c. Operasi atau metode adalah fungsi-fungsi yag dimiliki oleh suatu kelas.

Diagram kelas mendeskripsikan jenis-jenis objek dalam sistem dan berbagai hubungan statis yang terdapat diantara mereka. Diagram kelas juga menunjukan properti dan operasi sebuah kelas dan batasan-batsan yag terdapat dalam hubungan-hubungan objek tersebut.

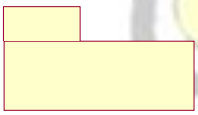
Diagram kelas menggambarkan struktur dan deskripsi *class*, *package* dan objek beserta hubungan satu sama lain seperti *containment*, pewarisan, asosiasi, dan lain-lain.

Kelas memiliki tiga area pokok :

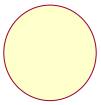
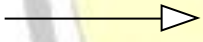
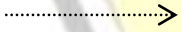
a. Nama, merupakan nama dari sebuah kelas
b. Atribut, adalah karakteristik data yang dimiliki suatu objek dalam kelas
c. Operasi, adalah fungsi atau transformasi yang mungkin dapat diaplikasiakan ke atau oleh suatu objek dalam kelas.

Berikut adalah Simbol-simbol pada *class diagram* seperti pada tabel 2.5 [7].

Tabel 2.5 : Simbol *Class Diagram* [Sumber: 7]

Simbol	Nama	Deskripsi
	*Package*	Merupakan sebuah bungkusan dari satu atau lebih kelas

Tabel 2.5 (Lanjutan)

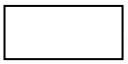
	Operasi	Kelas pada struktur sistem
	Antarmuka atau *Interface*	Sama dengan konsep *interface* dalam pemrograman berorientasi objek
	*Asosiasi*	Relasi antar kelas dengan makna umum, *asosiasi* biasanya juga disertai dengan *multiplicity*
	Asosiasi berarah atau *Directed Asosiasi*	Relasi antar kelas dengan makna kelas yang satu digunakan oleh kelas yang lain, *asosiasi* biasanya juga disertai dengan *multiplicity*
	*Generaliasasi*	Relasi antar kelas dengan makna *generalisasi-spesialisasi* (umum khusus)
	Kebergantungan atau *Defedency*	Relasi antar kelas dengan makan kebergantungan antar kelas
	*Agresasi*	Relasi antar kelas dengan makna semua–bagian (*whole-part*)

2.3.2.3 Sequence Diagram

Diagram sequence menggambarkan kelakuan atau perilaku objek pada *use case* dengan mendeskripsikan waktu hidup objek dan message yang dikirm dan diterima antar objek. Oleh karena itu untuk menggambarkan *diagram sequence* maka harus

diketahui objek-objek yang terlibat dalam sebuah *use case* beserta metode-metode yang dimiliki kelas yag diinstansiasi menjadi objek itu. Berikut adalah Simbol-simbol pada *Sequence Diagram* seperti pada tabel 2.6 [7].

Tabel 2.6 : Simbol *Sequence Diagram* [Sumber: 7]

Gambar	Nama	Keterangan
	*Actor*	*Actor* menggambarkan segala pengguna *software* aplikasi (*user*).
	*Object*	*Object* merupakan bagian untuk suatu *actor* mengambarkan sesuatu.
	Object Message	Garis untuk memberikan pengertian berupa pesan kepada objek.
	*Time*	Merupakan waktu yang dibutuhkan didalam obyek tersebut.
	Llifeline	Untuk mengurutkan pesan dari atas ke bawah.
	Panggilan Mandiri	Digunakan untuk menampilkan data yang sudah ada atau sudah tersimpan sebelumnya.

2.3.2.4 Activity Diagram

Diagram aktivitas atau *Activity Diagram* menggambarkan *workflow* (aliran kerja) atau aktivitas dari sebuah sistem atau proses bisnis. Yang perlu diperhatikan disini adalah bahwa diagram aktivitas menggambarkan aktivitas sistem bukan apa yang dilakukan aktor, jadi aktivitas yag dapat dilakukan oleh sistem. Diagram aktivitas mendukung perilaku paralel.

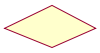
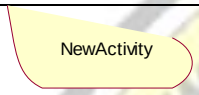
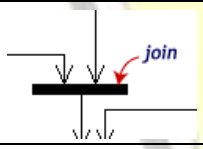
Diagram aktivitas juga banyak digunakan untuk mendefinisikan hal-hal berikut:

a. Rancangan proses bisnis dimana setiap urutan aktivitas yang digambarkan merupakan proses bisnis sistem yang didefinisikan.
b. Urutan atau pengelompokan tampilan dari sistem atau *user interface* dimana setiap aktivitas dianggap memiliki sebuah rancangan antar muka tampilan.

c. Rancangan pengujian dimana setiap aktivitas dianggap memerlukan sebuah pengujian yang perlu didefinisikan kasus ujiannya.

Berikut adalah Simbol-simbol pada *Activity Diagram* seperti pada tabel 2.7 [7].

Tabel 2.7 : Tabel Simbol *Activity Diagram* [Sumber: 7]

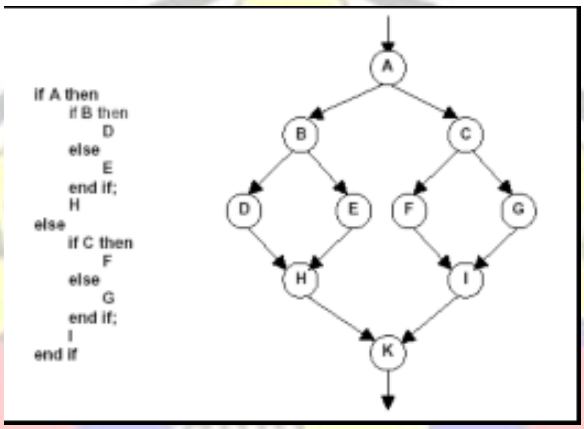
Gambar	Nama	Keterangan
	*Start*	Untuk memulai suatu *activity diagram*
	Kondisi	Untuk menjelaskan pernyataan lanjut atau tidak.
	*Activity*	Untuk Menggambarkan aktivitas yang ada.
	*Forking*	Satu aliran yang pada tahap tertentu berubah menjadi beberapa aliran
	Interaksi	Interaksi adalah suatu perilaku yang mencakup himpunan pesan-pesan (*message*) yang diperlukan untuk menyelesaikan suatu fungsi tertentu.
	*End*	Untuk mengkhiri suatu *activity diagram*.
	*Join*	Beberapa aliran sekaligus yang secara bersamaan masuk menjadi satu titik.

2.3.2.5 Collaboration Diagram

Collaboration diagram juga menggambarkan interaksi antar objek seperti *sequence diagram*, tetapi lebih menekankan pada peran masing-masing objek dan bukan pada waktu penyampaian message. Setiap *message* memiliki *sequence number* dimana *message* dari *level* tertinggi memiliki nomor 1 (satu). Message dari *level* yang sama memiliki prefiks yang sama [7].

2.4 Pengujian Sistem

Pengujian akan dilakukan dengan metode *White-box Testing* yaitu sebuah metode perancangan yang menggunakan struktur kendali dari desain prosedural. Teknik *White-box Testing* menggunakan alur logika dari program untuk membuat *test cases.* Alur logika suatu program dapat direpresentasikan dengan *flowgraph*, mencakup semua bagian dari program tertentu yang dieksekusi saat menjalakan program. Contoh *flow graph* dapat dilihat pada gambar 2.2 [8].



Gambar 2.2 : Contoh *flow graph* dari suatu kode program

Suatu *flow graph* terbentuk dari:

1. ***Nodes*** (titik), mewakili pernyataan (sub program) yang akan ditinjau saat eksekusi program.
2. ***Edges*** (anak panah), mewakili jalur alur logika program untuk menghubungkan satu pernyataan (sub program) dengan yang lainnya.
3. ***Branch nodes*** (titik cabang), titik-titik yang mempunyai lebih dari satu anak panah keluaran.
4. ***Branch edges*** (anak panah cabang), anak panah yang keluar dari suatu cabang.
5. ***Paths*** (jalur), jalur yang mungkin untuk bergerak dari satu titik ke lainnya sejalan dengan keberadaan arah anak panah [8]

.

Setelah *flow graph* selesai dibuat, kemudian *flow graph* tersebut diukur kompleksitasnya dengan menggunakan *cyclomatic complexity (CC).* Nilai yang dihitung menentukan jumlah jalur-jalur yang *independent* dalam kumpulan basis suatu program dan memberikan jumlah tes minimal yang harus dilakukan untuk memastikan bahwa semua pernyataan telah dieksekusi minimal satu kali [8]. Rumus penghitungan CC yaitu:

[Region / *Complexity*] V(G) = E (*edges*) – N (*nodes*) + 2

Contoh *flow graph* pada Gambar 3.2, diperoleh perhitungan:

V(G) = 11 – 9 + 2 = 4

Jalur 1 : 1-11

Jalur 2 : 1-2-3-4-5-10-1-11

Jalur 3 : 1-2-3-6-7-9-10-1-11

Jalur 4 : 1-2-3-6-8-9-10-1-11

2.5 *Tools* Yang Digunakan

2.5.1 *Google Maps* API

Google Maps merupakan suatu peta yang dapat dilihat dengan menggunakan suatu *browser*. Kita dapat menambahkan fitur *Google Maps* dalam web yang telah kita buat atau pada blog kita yang berbayar maupun gratis sekalipun dengan *Google Maps* API. *Google Maps* API adalah suatu *library* yang berbentuk *JavaScript*.[3]

API adalah kependekan dari *Application programming interface*. Dengan bahasa yang lebih sederhana, API adalah fungsi-fungsi pemrograman yang disediakan oleh aplikasi atau layanan agar layananan tersebut bisa di integrasikan dengan aplikasi yang kita buat. Jadi *Google maps* API adalah fungsi-fungsi pemrograman yang disediakan oleh *Google maps* agar *Google maps* bisa di integrasikan kedalam *Web* atau aplikasi yang sedang dibuat.[4]

---

[3] http://blog.xinthinx.us/2010/06/pengertian-google-maps-api.html (diakses pada tanggal 23 Februari 2015 pada jam 13.00 WIB)

[4] http://www.candra.web.id/2012/09/27/pengantar-google-maps-api/ (diakses pada tanggal 23 Februari 2015 pada jam 14.00 WIB)

2.5.2 Node.js

Node-JS adalah sebuah *platform software* yang dipakai untuk membangun aplikasi-aplikasi *serverside* yang fleksibel di sebuah *network* atau sebuah jaringan. *Node-JS* menggunakan *JavaScript* sebagai bahasa pemrograman dan dapat dengan mudah *throughtput* atau pemprosesan tingkat tinggi melalui *non-blocking* I/O. *Node-JS* memiliki fitur *built-in HTTP server library* yang menjadikannya mampu menjadi sebuah *web server* tanpa bantua *software* lainnya, seperti Apache atau Nginx. *Node-JS* dikembangkan berdasarkan teknologi *Google V8 JavaScript engine* serta berisi kompilasi skrip inti dan banyak modul siap pakai yang yang bermanfaat sehingga pengguna (dalam hal ini *web developer*) tidak perlu melakukan *coding* dan mendesain segalanya dari awal.[5]

Dengan memanfaatkan kekuatan dan kesederhanaan *Javascript*, pembuatan aplikasi *server-side* menjadi lebih mudah. *Javascript* merupakan Bahasa pemrograman yang paling banyak digunakan *programmer* web untuk menampung fungsi-fungsi yang akan dijalankan di *browser*. Untuk membangun *project* yang lebih kompleks, *programmer* dapat dengan mudah memilih, menginstal, dan menggunakan beberapa modul pihak ketiga. Semua perintah pada Node dijalankan dengan *Command-Line Interface* (CLI). Sebagai contoh, membuat kalimat *"Hello World!"* pada node dapat dilakukan dengan langkah sebagai berikut:

1. Menulis perintah:

```
console.log('Hello World!');
```

2. Menyimpan file dengan nama hello_world.js
3. Menjalankan dengan perintah CLI:

   node hello_world.js
4. Menghasilkan output kalimat : Hello World!
5. Untuk mengakhiri perintah CLI dapat dengan cara menekan ctrl+D atau ctrl+C.

Dalam *node* disediakan *package manager* yang bernama NPM untuk membantu mengintall modul yang diinginkan *user*. Untuk menginstal modul dapat

[5] http://www.ngulikweb.com/internet/mengenal-node-js-jalankan-javascript-di-server/ (diakses pada tanggal 24 Februari 2015 pada jam 10.00 WIB)

dilakukan dengan mengetikkan perintah *npm install*. Perintah dasar dari npm dapat dilihat pada tabel 2.8 [9].

Tabel 2.8 : Perintah dasar npm [Sumber: 9]

Perintah	Keterangan
npm install <package name>	Menginstal *package* yang didalamnya memuat folder yang bernama *node-module* sebagai modul pelengkap membangun suatu *web*.
npm uninstall -g <package name>	Menghapus *package* secara keseluruhan.
npm update -g <package name>	Memperbaharui *package* secara keseluruhan.
npm start	Menjalankan *file* utama dari Node.JS yang bernama app.js sebagai *file index* dari *project* yang sedang dibangun.

2.5.3 MongoDB

Mongodb merupakan salah satu jenis *database* dari NoSQL. Meski namanya aneh tapi kehandalannya sudah dibuktikan oleh website-website dengan *big trafic*, dan tentunya *big database*, seperti Foursqure, Disney, Forbes, Sourceforge, ataupun Github. Mongo sebetulnya *database* yang tidak memiliki relasional, berbeda dengan MySQL yang memang dibuat untuk menangani relasi database. Justru mongo memang dibuat menggunakan manajemen database berorientasi *document oriented*.[6]

Pada MongoDB juga terdapat *Data Manipulating Language* (DML) yaitu seperangkat perintah yang dimiliki oleh *database* yang meliputi *Create* atau *Insert, Update, Delete*, serta *Read* atau *Select* atau sering disebut (CRUD). Perintah DML dalam MongoDB yaitu:

[6] http://desainweb.ilmuwebsite.com/2014/10/nodejs-mongodb-masa-depan-dunia-web.html (diakses pada tanggal 24 Februari 2015 pada jam 12.00 WIB)

1. *Insert*

```
server.GetDatabase("[db]").GetCollection("[table]")
        .Insert(object)
```

2. *Update*

```
server.GetDatabase("[db]").GetCollection("[table]")
        .Update(Query, Update.Set("field"
```

3. *Delete*

```
server.GetDatabase("[db]").GetCollection("[table]")
        .Delete (Query)
```

4. *Select*

```
server.GetDatabase("[db]").GetCollection("[table]")
        .Find (Query)
```

Dapat dilihat dari *syntax* yang telah dipaparkan sebelumnya, bahwa proses *query* dengan menggunakan MongoDB memiliki kemiripan dengan query SQL Server.[10]

Mongodb menyimpan data dalam bentuk *collection* berupa struktur tipe data seperti *array*. Collection dalam MongoDB menggunakan struktur data J-SON. Perintah dasar yang biasa digunakan dalam MongoDB seperti pada tabel 2.3 [11].

Tabel 2.9 : Perintah dasar MongoDB [Sumber: 11]

Perintah	**Keterangan**
Use <nama_db>	Membuat database baru atau memilih database yang sudah ada untuk digunakan.
Db.mycoll.save(object)	*Insert* data baru ke dalam collection
Db.mycoll.update(kondisi)	*Update* data yang ada dalam collection
Db.mycoll.remove(kondisi)	*Delete* data yang ada dalam collection
Db.mycoll.find(kondisi)	*Select* ata mencari data yang ada dalam collection.

2.5.4 NoSQL

NoSQL adalah sebuah konsep mengenai penyimpanan data non-relasional. Berbeda dengan model basis data relasional yang selama ini digunakan, NoSQL menggunakan beberapa metode yang berbeda. Metode ini bergantung dari jenis database yang digunakan. Karena NoSQL sendiri merupakan konsep database, dan pada implementasinya, banyak jenis-jenis dari NoSQL ini. Dalam NoSQL, terdapat berbagai macam bentuk database dengan skema dan konsepnya sendiri. Sama halnya dengan database SQL seperti MySQL, Oracle dan lain-lain. Dalam NoSQL terdapat MongoDB, CouchDB dan lain-lain.[7]

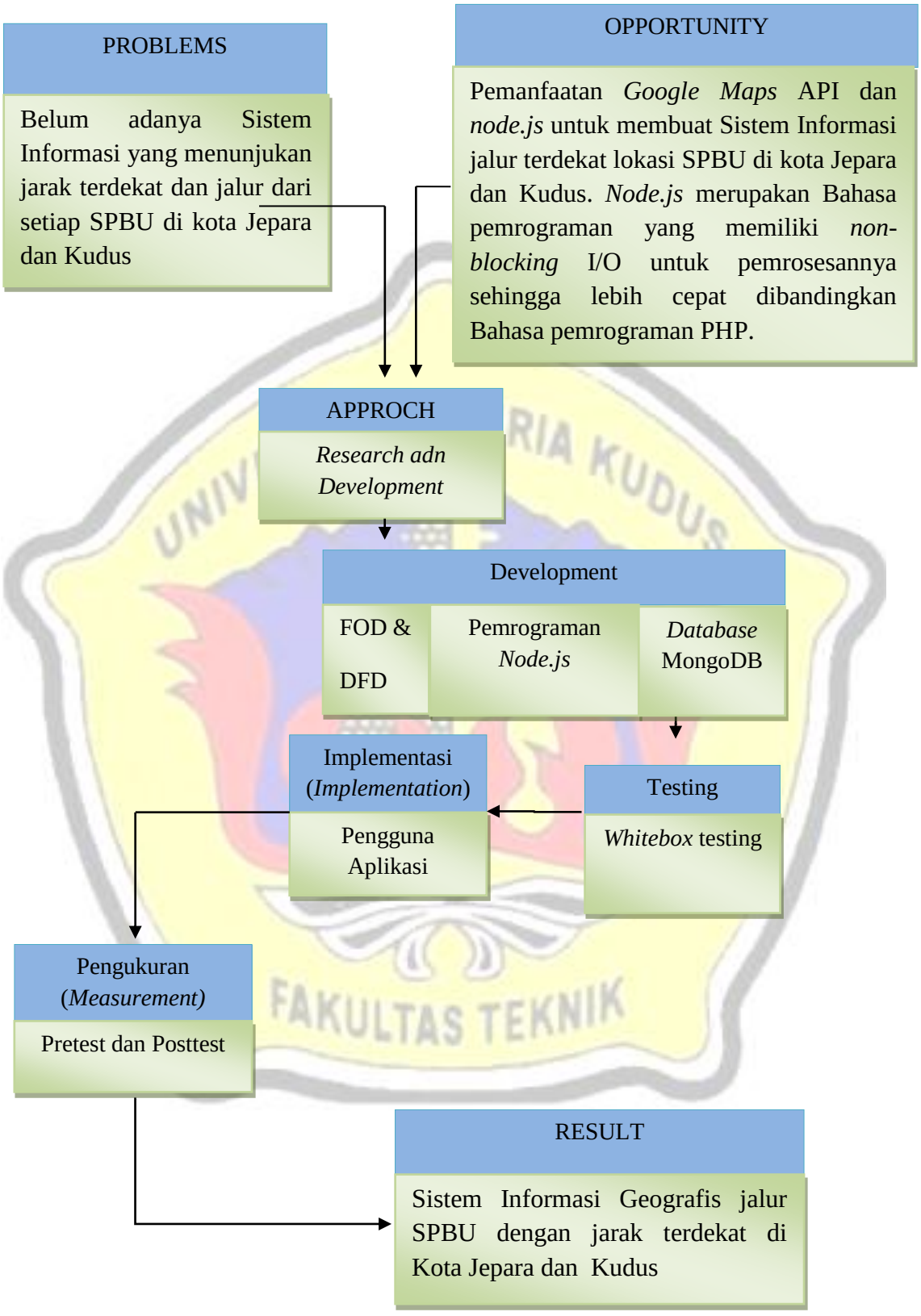
2.5.5 Sublime Text 3

Sublime Text 3 adalah sebuah *text/source editor* yang memudahkan kita saat melakukan *coding* dan *support* terhadap banyak bahasa pemrograman mulai dari ASP, C, C#, C++, PHP, HTML, dan masih banyak lagi. Sublime Text 3 ini juga hadir dengan berbagai macam tema yang membuat text *editor* ini terlihat lebih indah dan nyaman saat melakukan coding. Sublime Text 3 juga tersedia di berbagai sistem operasi seperti Windows, Linux, dan Mac OS.[8]

[7] http://blog.randisunarsa.web.id/?p=383 (di akses pada tanggal 24 Februari 2015 pada jam 20:00 WIB)

[8] http://msfkonsole.blogspot.com/2014/11/cara-install-sublime-text-3-license-key.html (diakses pada tanggal 7 Maret 2015 pada jam 13.00 WIB)

2.5 Kerangka Pemikiran



Gambar 2.3 : Kerangka Pemikiran